Modulprüfung Programmieren II

Wintersemester 2004 / 05 (Beispiel-Klausur)

Datum:	
Beginn:	09:15 Uhr
Dauer:	90 Minuten

Matrikelnr.:	
Name, Vorname:	

Maximale Punktzahl:	90 Punkte
Erreichte Punkte	
Note:	

Aufgabe 1 (15 Punkte)

Gegeben ist die Klasse "**Vektor2D**":

```
class Vektor2D
{
private:
    float x, y;
public:
    Vektor2D(float v_x, float v_y);

    void  print();    // Ausgabe des Vektors auf der Konsole in der Form "x | y"
    float abs();      // Laenge des Vektors

    static Vektor2D   add(Vektor2D a, Vektor2D b);   // Vektoraddition von a und b
    static Vektor2D   sub(Vektor2D a, Vektor2D b);   // Vektorsubtraktion von a und b
    static float    ska_p(Vektor2D a, Vektor2D b);   // Skalarprodukt der Vektoren a und b
};
```

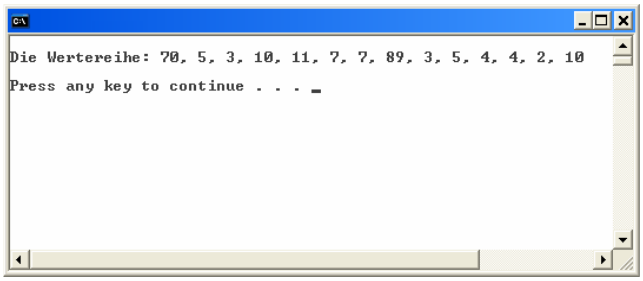
Erweitern Sie diese Klasse um die statische Methode "**float angle_d(Vektor2D a, Vektor2D b)**", welche den Winkel (0° bis 180°) zwischen den beiden übergebenen Vektoren berechnen soll! Verwenden Sie dazu die anderen Methoden der Klasse und entsprechende Funktionen aus der Bibliothek "**math**", wie im **Anhang A.1** beschrieben. Der Anhang liefert zudem den Zusammenhang zwischen dem Winkel zweier Vektoren und deren Skalarprodukt.

Aufgabe 2 (20 Punkte)

Schreiben Sie eine Funktion "**`void arrayprint(T* a, int n)`**", welche die Elemente eines beliebigen Arrays auf der Konsole darstellt. Das "**`T`**" im Funktionskopf steht für einen beliebigen Datentyp.
Als Parameter soll der Funktion zum einen das Array und zum anderen die Anzahl der auszugebenen Zeichen (bzw. die Länge des Arrays) übergeben werden.

Ein einfaches Programm, das diese Funktion verwendet, kann folgendermaßen aussehen:

```
int main(int argc, char *argv[])
{
    int werte[] =
        {70,5,3,10,11,7,7,89,3,5,4,4,2,10};
    cout << "Die Wertereihe: ";
    arrayprint(werte, 14);
    cout << endl;
    return 0;
}
```

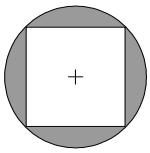


Hinweis: Verwenden Sie die Möglichkeit in C++, Template-Funktionen zu erstellen.

Aufagbe 3 (30 Punkte)

a) (15 Punkte)

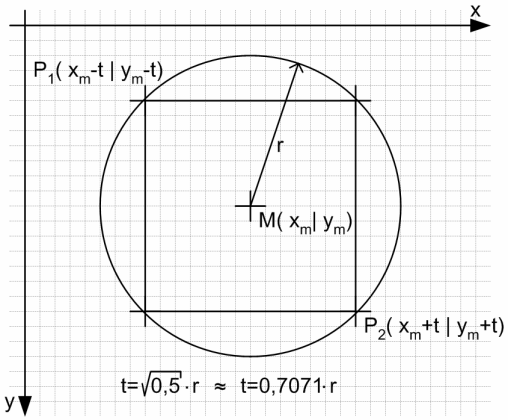
Gegeben sind die – ihnen aus der Übung bekannten – Klassen "`figure`" (*dt. Figur*), "`circle`" (*dt. Kreis*) und "`rectangle`" (*dt. Rechteck*), welche geometrische Objekte beschreiben. Der **Anhang A.2** zeigt den Aufbau aller drei Klassen inklusive der Methodenprototypen. Ihre Aufgabe ist es, die **Schnittstelle** (d.h. nur **die Prototypen der Methoden)** einer neuen Klasse "`pattern2`" (*dt. Schablone2*) zu erstellen, welche sich aus den Klassen "`circle`" und "`rectangle`" ableitet.



Die Abbildung oben veranschaulicht diese Klasse "`pattern2`" grafisch. Ein Objekt dieser Klasse stellt demnach eine Schablone dar, dessen Erscheinung durch einen Kreis gekennzeichnet ist, aus dem ein möglichst großes Quadrat ausgeschnitten wurde.

b) (15 Punkte)

Schreiben sie nun den Konstruktor der abgeleiteten Klasse "`pattern2`", dessen reinen Aufbau sie oben entwickelt haben. Die folgende Abbildung zeigt den Zusammenhang zwischen Mittelpunkt und Radius des Kreises und den beiden charakteristischen Eckpunkten des innen liegenden Quadrates.



Beispiel: Das Quadrat, welches in einem Kreis mit dem Mittelpunkt M(5|5) und dem Radius r=1,4142 liegt, hat folgende Eckpunkte: $P_1(4|4)$ und $P_2(6|6)$.

Aufgabe 4 (15 Punkte)

Gegeben sei die Klasse

```
class animal {
   private:
      char *name;
      int legs;
   public:
      animal();
      animal(char *my_name, int legs);
}
```

Erläutern Sie an diesem Beispiel den Unterschied zwischen der Erzeugung von statischen und dynamischen Objekten. Erzeugen Sie die Objekte „**cat**“ und „**dog**“ statisch sowie die Objekte „**horse**“ und „**cow**“ dynamisch. Ergänzen und kommentieren Sie die entsprechenden Programmzeilen so, dass der Unterschied deutlich wird.

Aufgabe 5 (10 Punkte)

Entwerfen Sie den Prototyp eines Konstruktors für eine Klasse „**queue**“ (dt. Schlange) mit den folgenden Eigenschaften:

- Zur Identifikation erhält jede „**queue**“ einen eindeutigen Namen (char*).
- In einer „**queue**“ können beliebig viele **float**-Werte abgespeichert werden.
- Falls nicht anders angegeben, soll ein „**queue**“-Objekt maximal 50 Elemente aufnehmen können.

A. Anhang:

A.1 Zu Aufgabe 1

Der Winkel zwischen zwei Vektoren

Um den Winkel $\varphi = \sphericalangle(\vec{a}, \vec{b})$ zwischen zwei Vektoren $\vec{a}$ und $\vec{b}$ zu bestimmen, kann das Skalarprodukt ($\vec{a} \cdot \vec{b}$) verwendet werden:

$$\vec{a} \cdot \vec{b} = |\vec{a}| \cdot |\vec{b}| \cdot \cos \sphericalangle(\vec{a}, \vec{b})$$

$$\Rightarrow \sphericalangle(\vec{a}, \vec{b}) = \arccos\left(\frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \cdot |\vec{b}|}\right)$$

Mathematische Funktionen in der Bibliothek "math"

Der folgende Ausschnitt zeit die notwendigen Zeilen aus der Header-Datei "math.h":

```
   :
#define M_PI 3.14159265358979323846 // Kreiszahl PI
   :
float acos (float);                 // Arcuscosinus: arccos() Ergebnis in RAD
   :
```

Hinweis: Der Rückgabewert der Funktion "**acos()**" ist ein Winkel im Bogenmaß (RAD). Um einen Wert in Grad (0° bis 180°) zu erhalten, muss dieses Ergebnis noch entsprechend skaliert werden.Hierbei gilt:

(Winkel in Grad) = 180 * (Winkel im Bogenmaß) / π

A.2 Zu Aufgabe 2

```
class figure
{
    private:
        char* name;
        color paint;
        bool  visible;

    public:
        figure(char* n, color c, bool v);
        ~figure();

        color get_color();
        bool  get_visible();
        char* get_name();

        virtual float calc_surface() = 0;
        virtual void display() = 0;
};


class circle : public figure
{
    protected:
        float x, y, r;           // Koordinaten des Mittelpunktes und der Radius

    public:
        circle(char* n, color c, bool v, float pos_x, float pos_y, float radius);
        ~circle();

        float calc_surface();    // Oberfläche berechnen
        void display();          // Eigenschaften auf der Konsole darstellen
};


class rectangle : public figure
{
    protected:
        float ax, ay, bx, by;    // Die zwei charakteristischen Eckpunkte

    public:
        rectangle(char* n, color c, bool v, float p1x, float p1y, float p2x, float p2y);
        ~rectangle();

        float calc_surface();    // Oberfläche berechnen
        void display();          // Eigenschaften auf der Konsole darstellen
};
```